ISO/WD 18876-2

ISO TC184/SC4/WG10 N315 **Date: 2000-09-08**

Supersedes:

Industrial automation systems and integration — Integration of industrial data for exchange, access, and sharing — Part 2: Integration and mapping methodology

ABSTRACT:

This document defines the integration and mapping methodology component of the ISO 18876 architecture for integration of industrial data.

KEYWORDS:

industrial data, integration, exchange, access, sharing, architecture, overview

COMMENTS TO READER:

This is an incomplete draft of Part 2 of ISO 18876 for discussion at the WG10 meeting in Winter Park (2000-09-11/15). Editor's notes are included as boxed text; these include open issues as well as outline descriptions of the clauses whose detailed content has yet to be created. A second working draft, taking into account results of the Winter Park discussions, will be prepared for WG10 review at the Charleston meeting (2000-10-14/20).

Reviewers are encouraged to submit and discuss issues via the ISO 18876 list that has been established at egroups – see http://www.iso18876.org for details.

Project leader:	Julian Fowler PDT Solutions Belle Vue Barn, Mewith Lane Bentham Lancaster LA2 7DQ UK	Part editor:	Julian Fowler PDT Solutions Belle Vue Barn, Mewith Lane Bentham Lancaster LA2 7DQ UK
Telephone: Fax: Email:	+44 15242 63389 +44 870 052 3414 jfowler@pdtsolutions.co.uk	Telephone: Fax: Email:	+44 15242 63389 +44 870 052 3414 jfowler@pdtsolutions.co.uk

Document type:	International Standard	Document subtype:	Not applicable
Document stage:	Working Draft (20)	Document language:	E
File name:	iso18876-2_wd_01.doc	Template:	ISO SC4 latest.dot

ISO Copyright Office
Case Postale 56 • CH-1211 Genève 20 • Switzerland
Tel. + 41 22 749 01 11
Fax + 41 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Contents

Page

Figures

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75% of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO 10303 may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

International Standard ISO 18876-1 was prepared by Technical Committee ISO/TC184, *Industrial automation systems and integration*, Subcommittee SC4, *Industrial data*.

ISO 15926 consists of the following parts under the general title *Industrial automation systems and integration – Integration of industrial data for exchange, access, and sharing*:

— Part 1, Architecture overview and description;

— Part 2, Integration and mapping methodology.

The structure of this International Standard is described in ISO 18876-1.

Annex A forms a normative part of this part of ISO 18876. Annexes B, C, and D are provided for information only.

Introduction

Text to be added

Industrial automation systems and integration — Integration of industrial data for exchange, access, and sharing — Part 2: Integration and mapping methodology

1 Scope

The following is taken from the NWIP, with a change from "the integration model" to "integration models" or "an integration model"

This part defines methods for analysis of information requirements, creation and transformation of data models, and mapping between data models, satisfying the following requirements:

— creation, extension and updates to integration models;

One possible addition here is a method for *selection* of an integration model. For example, given two application data models A1 and A2, what criteria determine whether ISO 15926-2 is a suitable model for integrating them? How are these criteria then assessed (or is this simply a matter of *doing* the mapping/integrating and addressing limitations in the integration model when they are found? The key issue underlying this is whether ISO 18876 assumes *one* integration model, or can support development and use of *many* integration models within a common architectural framework.

— creation of views on an integration model to support particular application domain requirements for exchange and/or sharing;

— creation of mappings from external application models to an integration model.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO 15926. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 8824-1:1994, *Information technology — Open systems interconnection — Abstract syntax notation one (ASN.1) — Part 1: Specification of basic notation*.

ISO 10303-1:—[1)], *Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles.*

ISO 10303-11: —[2)], *Industrial automation systems and integration — Product data representation and exchange — Part 11: The EXPRESS language reference manual*.

[1)] To be published. (Revision of ISO 10303-1:1994)

ISO 18876-1: —[3], *Industrial automation systems and integration — Integration of industrial data for exchange, access, and sharing — Part 1: Architecture overview and description*.

Others to be added as required – I've assumed here that we will want to reference the second editions of Part 1 and Part 11 of STEP.

3 Terms, definitions, and abbreviations

For the purposes of this part of ISO 18876, the following terms, definitions and abbreviations apply; those taken or adapted from ISO 10303-1, ISO 10303-11, or ISO 18876-1 are repeated below for convenience.

NOTE Definitions copied verbatim from ISO 10303-1:1994 are followed by "[ISO 10303-1]". An explanatory note follows definitions that have been adapted from ISO 10303-1.

3.1
application data model
data model that represents information used for some particular purpose

[ISO 18876-1]

3.2
data
a representation of information in a formal manner suitable for communication, interpretation, or processing by human beings or computers

[ISO 10303-1]

3.3
data model

I assume that there must be a JTC1 terminology standard of some kind that might have a definition that we could use or at least start from.

3.4
data population
collection of data instances that conform to a specified data model

NOTE The size of the collection may be zero, in which case the data population is equivalent to the data model.

[ISO 18876-1]

This term introduced as a placeholder for the concept of model+instances

3.5
information
facts, concepts, or instructions

[ISO 10303-1]

---

[2] To be published. (Revision of ISO 10303-11:1994)

[3] To be published.

3.6
integrated data model
data model that is mapped to a subset of an integration data model

[ISO 18876-1]

3.7
integration
activity that creates, modifies, or extends an integration data model

[ISO 18876-1]

3.8
integration data model
data model that represents information in a non-redundant, sharable manner, and that is independent of any particular use of the information that it represents

[ISO 18876-1]

3.9
map
mapping specification
definition of the association between each element of a data population and elements of another data population

[ISO 18876-1]

NOTE 1 A map can be uni-directional or bi-directional.

NOTE 2 A map can include specification of data atructure transformations, data value transformations, and data encoding transformations.

3.10
mapping
activity that creates and documents a map between two data populations

[ISO 18876-1]

3.11
model
simplified, consistent description of selected aspects of something, used for a purpose

NOTE A model can be represented by data, or by a data model, or by a combination of these. See Annex B for further discussion of the relationship between models, data, and data models.

3.12
model representation

Further definitions to be added as required by the text

4 Methodology overview

Subsequent to writing this section I note that the overview given in Part 1 is possibly more detailed that what is given here. This will have to be discussed in Winter Park and a more sensible division between the rwo parts determined.

4.1 Purpose

This section was originally divided into subclauses on “Integration” and “Mapping”; however, describing these as separate activities is almost impossible. The approach now is to characterize “mapping” as the activities of describing the relationship between the elements of an integrated model and the corresponding subset of an integration model. At this level, mapping is then a sub-activity of the overall “integrating” process. The title of this subclause may not be appropriate – the distinction between 4.1 and 4.2 is that this presents an overview from an inputs/outputs perspective, whereas 4.2 focuses more on the process.

The methodology defined in this part of ISO 18876 is designed to meet the following requirements.

4.1.1 Integrating application data models

This component of the methodology integrates two application data models by creating an integration model that is capable of representing all of the concepts and constraints of the application data models. Such an integration model supports sharing of data between applications and users that operate using the application data models.

Assume that Part 1 does (or will) describe what is meant by “sharing” in this context – this needs to cover the role of the integration model as the basis for a data store in which data from the ADMs can be reconciled, and as a “mediator” for exchange of data between the two ADMs. The difference between these needs to be addressed by the “services” element of the IIDEAS architecture.

This requirement and its solution are illustrated in Figure 1 below. The inputs to the activity are as follows:

— two or more application data models;

NOTE 1 Although it is possible to integrate more than two application data models in one step, it is simpler to perform the integrating activity on two application data models first, and then extend the resulting integration model to satisfy the requirements of additional application data models.

— principles for the creation of an integration model.

NOTE 2 The use of such principles is an important criterion in determining the future extensibility and reuse of an integration data model.

The outputs of this activity are as follows:

— an integration data model that represents the semantics and constraints of the input application data models;

— maps that define the relationships between elements of each application data model and their corresponding subsets of the integration data model.

Recalling past discussions with Felix Metzger, I think that the use of subset here is correct, if only in the sense that the integration data model is defined by a set whose members are entity data type declarations (assuming an E-R paradigm) and/or entity instances – each ADM is related by a map to some (possibly) all of these declarations and instances.

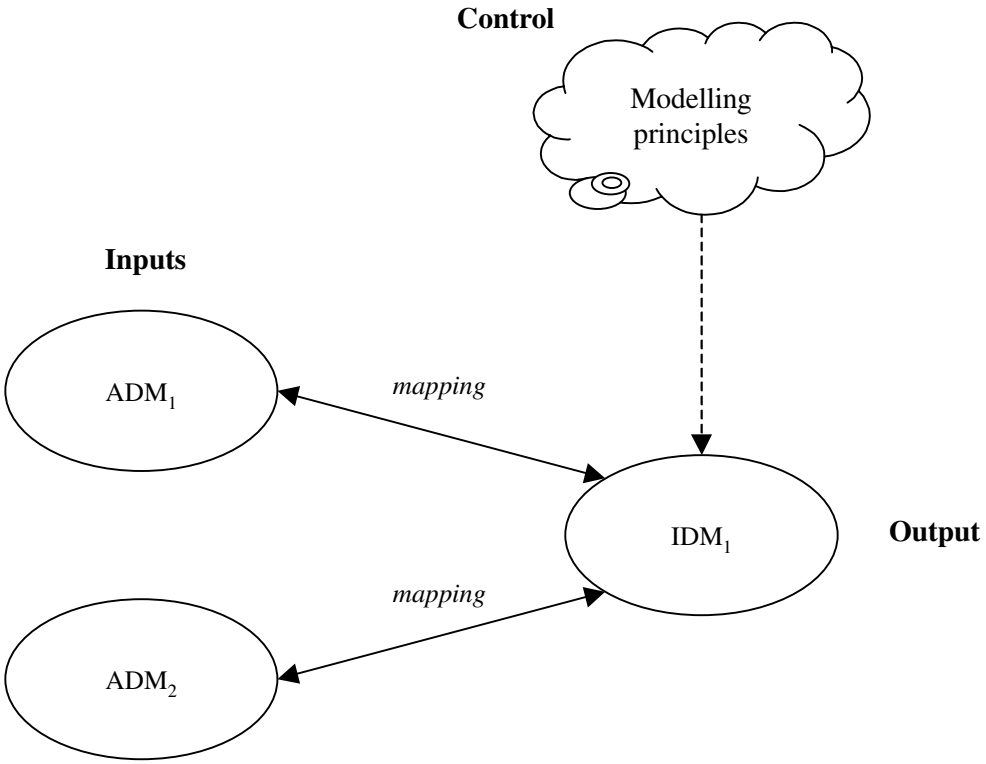

Figure 1 – Creating an integration data model from two application data models

4.1.2 Integrating an application data model and an integration data model

This component of the methodology integrates an application data model and an existing integration data model. The purpose of this activity may be either or both of the following:

— integrating the application data model with other integrated data models;

Apart from the definition in clause 3, is additional text needed here to characterize what an "integrated data model" is? Would it help to depict existing, integrated data models in Figure 2?

— improving the quality of the application data model by representing its concepts and constraints in a more consistent and extensible form and structure.

This requirement and its solution are illustrated in Figure 2 below. The inputs to the activity are as follows:

— an application data model;

— an existing integration data model;

— the modelling principles used to create the integration data model – these also determine how the integration data model is extended to meet the requirements of the application data model.

The outputs of the activity are as follows:

— an extended integration data model (if the input integration data model does not precisely satisfy the requirements of the application data model);

— a map that defines the relationships between elements of the application data model and the corresponding subsets of the extended integration data model.

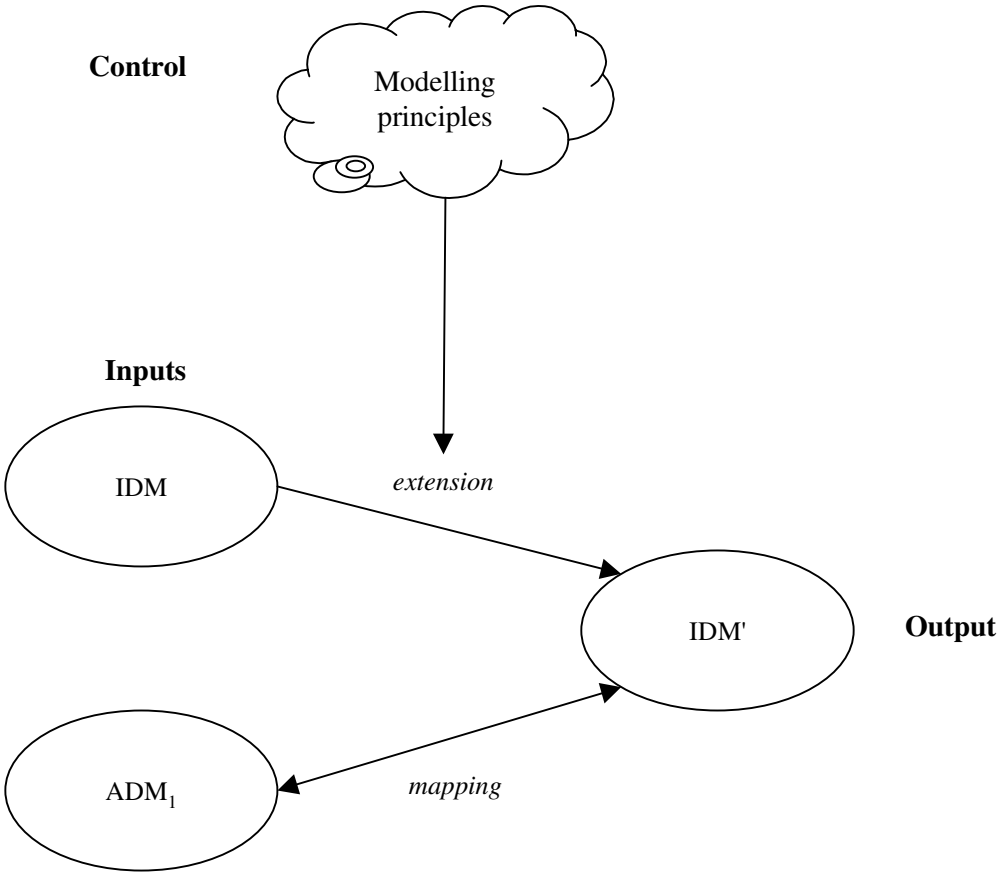

Figure 2 – Integrating an application data model with an existing integration data model

4.1.3 Defining an application data model as a view of an integration data model

This component of the methodology creates an application data model as a view onto an integration data model. The purpose of this activity is to enable use of the integration data model for sharing data between applications or a community of users that share common information requirements.

This requirement and its solution are illustrated in Figure 3 below. The inputs to the activity are as follows:

— a defined collection of information requirements which may be described by one or more of the following:

  — existing data;

  — existing data models;

  — usage scenarios;

  — paper documents and forms;

  — results of interviews with users;

  — existing applications.

— an existing integration data model;

— the modelling principles used to create the integration data model – these also determine how the integration data model is extended to meet the stated information requirements.

The outputs of the activity are as follows:

— an extended integration data model (if the input integration data model does not precisely satisfy the stated information requirements);

— an integrated data model that satisfies the stated information requirements;

NOTE The integrated data model may use the structure and/or the terminology of the integration data model. If it uses both, then it is identical to a subset of the integration data model, and the map between them is trivial.

— a map that defines the relationships between elements of the integrated data model and the corresponding subset of the extended integration data model.

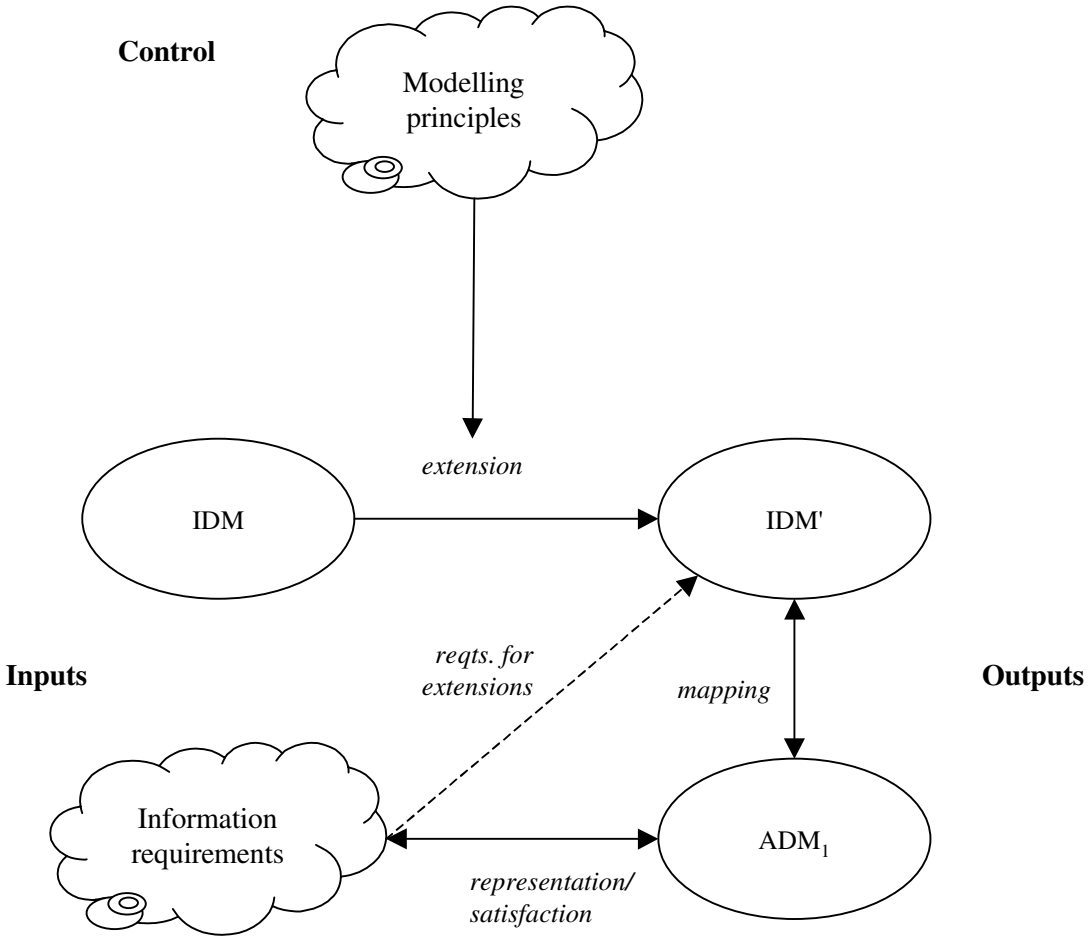

Figure 3 – Creating an application data model as a view on to an integration data model

4.2 Process overview

This clause presents an overview of the integrating and mapping activities. These are divided into four stages:

— analysis of the application data model;

— extension of the integration data model (if required);

— identification of the subset of the integration data model that corresponds to the application data model;

— definition of a map between the application data model and the identified subset of the integration data model.

These activities all assume existence of an application data model – at least one further section is needed for the scenario of creating an IDM from two ADMs (if we agree that this should be in scope).

4.2.1 Analysis of the application data model

The purpose of this activity is to determine how the concepts that are represented by an application data model related to the concepts of the chose integration data model. This analysis is illustrated in Figure 4 below.

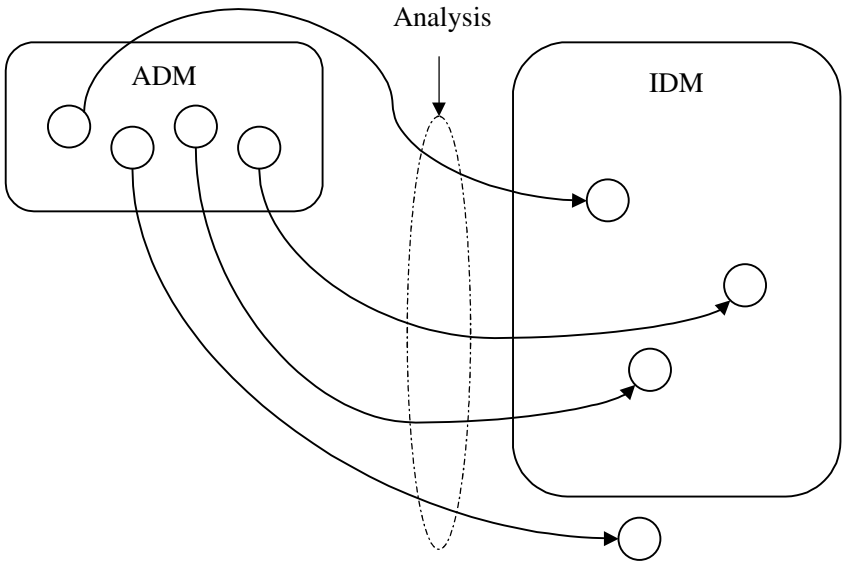

Figure 4 – Analysis of application data model

The nature of these relationships varies considerably, depending on the nature of the models involved. Some or all of the following relationship types may be recognized during this analysis:

— equivalence: the same concept exists in the application data model and the integration data model, and has the same name in both cases;

— synonyms: the same concept exists in the application data model and the integration data model, but the models use different names;

— homonyms: concepts exist within the application data model and the integration data model that have the same name but have different meanings;

— complex entity data types (in the application data model): a single concept in the application data model corresponds to more than one concept in the integration data model;

EXAMPE 1 An application data model includes an entity data type called **product** which can represent either individual manufactured items (identified by serial numbers) or classes of manufactured items (identified by part numbers). If these are recognized as separate concepts in the integration data model as **physical_object** and **class_of_physical_object**, then instances of **product** in the ADM can correspond to instances of both concepts in the IDM.

— partitioned entity data types (in the application data model): two or more concepts in the application data model correspond to a single general concept in the integration model.

EXAMPLE 2 An application data model contains entity data types called **customer** and **supplier**. These may both correspond to a general entity data type in the integration data model called **organization**. If the IDM does not include

the necessary constructs to distinguish between organizations playing the roles of ‘customer’ and ‘supplier’ then these will have to be added during the extension activity.

Obviously this is not a comprehensive list – the issue here is how much further can this go without getting into the specifics of a particular integration model and/or modelling language? In particular, what should we (can we) say here about handling of attributes and relationships – see mapping section for further discussion of this.

Lastly, it is possible that a concept within the application data model has no equivalent within the integration data model. In this case it is necessary to develop extensions to the integration data model so that a subset of the IDM with precise semantic equivalence to the ADM can be identified.

The output of the analysis activity is documentation of the semantic relationships that have been discovered between the concepts of the application data model and the integration data model, together with identification of any voids that have been discovered in the integration data model.

4.2.2 Extending the integration data model

If the analysis activity described in 4.2.1 discovers that one or more of the concepts represented in the application data model have no precise equivalent in the integration data model, then the latter shall be extended before the subsetting and mapping activities can be completed. This activity is illustrated in Figure 5 below.

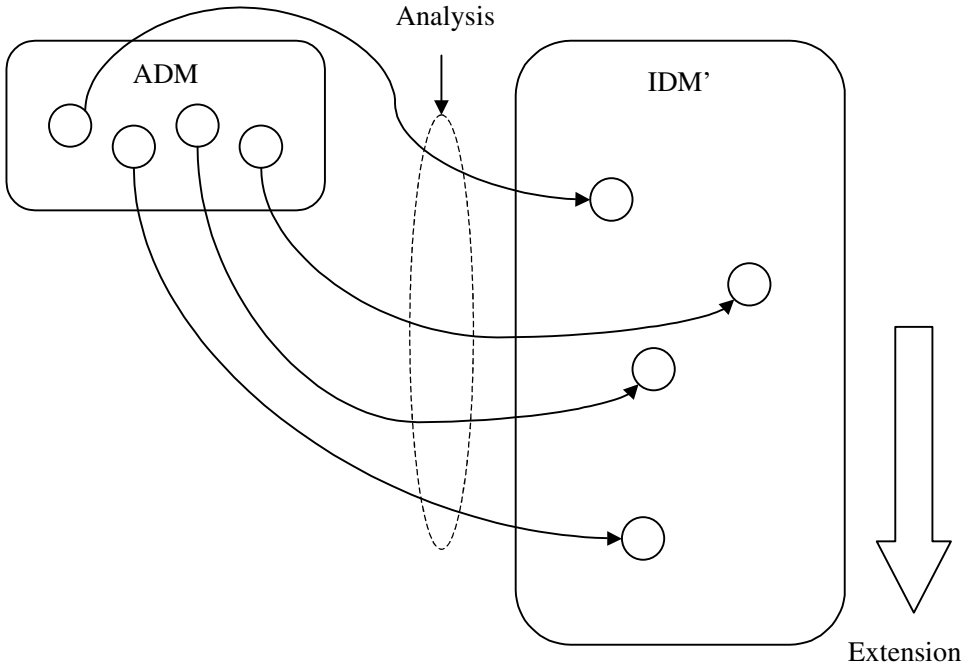

Figure 5 – Extending the integration data model

The method applied to extending a integration data model depends on the following:

— the structure and semantics of the existing integration data model;

— the modelling principles applied in its development and extension;

— the modelling language in which it is defined.

Subject to these dependencies, extension to an integration data model may have one or more of the following characteristics:

— creation of additional entity data types as specializations of existing entity data types in the integration data model;

— creation of standard entity instances within the integration data model.

Such extensions shall not modify existing entity data types in the integration data model since to do so can invalidate existing maps between the IDM and other ADMs.

NOTE 1 A well-designed IDM should not require such modifications to existing entity data types in response to new requirements.

NOTE 2 If there appears to be no choice but to modify an existing entity data type then the resulting extended IDM should contain both the unmodified and the modified entity data type so that all maps remain valid. The relationship between the modified entity data type and the unmodified one should be represented within the IDM, either as a structural relationship (usually specialization) or by using an internal map.

I introduced the following distinction in a presentation given at the Melbourne meeting. I'm still not sure that it is valid; however, I've retained it here so that it can be discussed in further detail.

Extensions to an integration data model can be characterized as follows:

— scope extension - requirements are discovered for data that is not covered at all by the integration data model;

— detail extension – the integration data model does not include constructs that precisely match the required semantics.

4.2.3 Identifying a subset of the integration data model

The purpose of this activity is to identify the subset of the integration data model that precisely corresponds to the application data model. In order for maps between the IDM and multiple ADMs to be managed, the identification of each ADM-specific subset shall form a part of the integration data model. The integration data model shall therefore include constructs that allow subset identifications to be recorded.

EXAMPLE If the IDM and ADMs of interest are defined in EXPRESS, the integration data model subset can be represented by a SCHEMA in which the selected entity data types of the integration data model are identified by USE FROM statements.

This may not be strictly necessary – especially if EXPRESS-X provides a similar capability in its VIEW and/or MAP constructs.

This activity is illustrated in Figure 6 below.

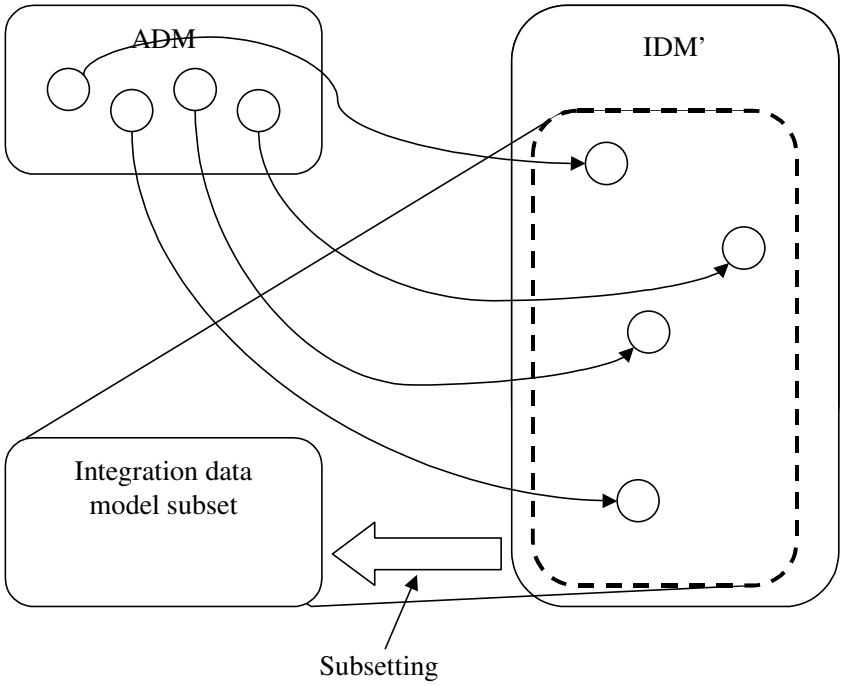

Figure 6 – Identifying a subset of the integration data model

In many cases the subsetting and mapping activities are iterative, in that the extent of the integration data model subset may not be fully determined until all the elements of the application data model have been mapped.

4.2.4 Mapping between the application data model and the identified integration data model subset

The purpose of this activity is to document, in an unambiguous and computer-interpretable form, all the relationships between elements of an application data model and the corresponding subset of an integration data model. The activities described above of analysis (see 4.2.1) and subsetting (see 4.2.3) establish a data specification (the IDM subset) that is semantically equivalent to the application detail model. This IDM subset will, however, in the general case differ from the application data model in its structure and/or its terminology. The mapping activity therefore establishes and describes the transformations that are applied to data that conforms to the IDM subset such that it conforms to the ADM (and vice versa). The mapping activity is illustrated in Figure 7 below.

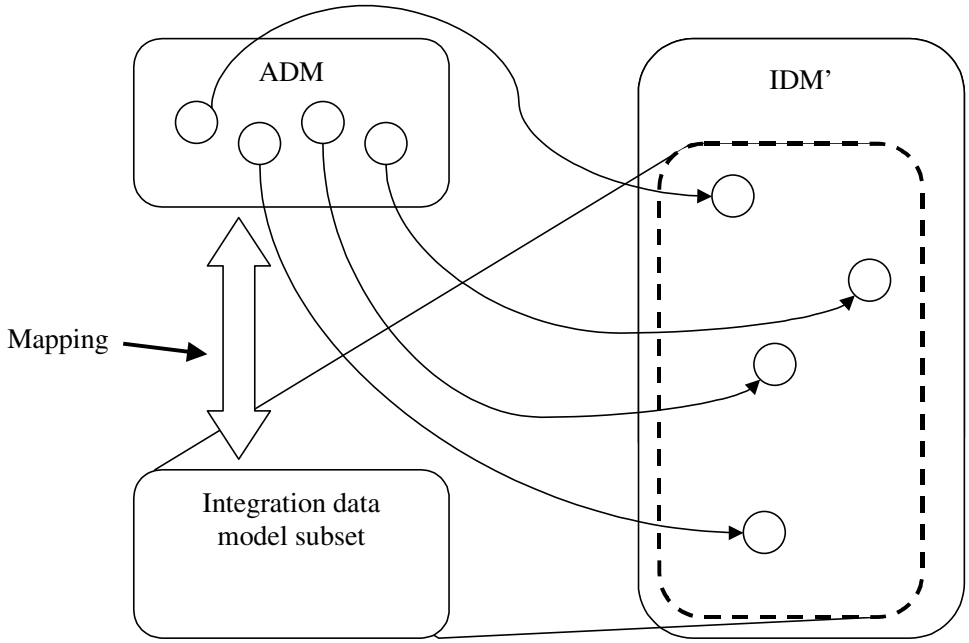

Figure 7 – Mapping between the application data model and the integration data model subset

The result of this activity is a map that characterizes the following:

— the correspondence of each entity data type, attribute, and relationship in the application data model with constructs in the identified subset in the integration data model subset;

— the correspondence of each entity data type, attribute, and relationship in the integration data model subset with constructs in the identified subset in the application data model.

The following constraints apply to these components of the map:

— every entity data type, attribute, and relationship in the application model shall have a defined correspondence with one or more constructs of the integration data model subset;

— every entity data type, attribute, and relationship in the integration data model subset shall have a defined correspondence with one or more constructs of the application data model or constraints on instantiation shall be specified in the context of this map.

The intent of this second constraint is to allow for the case in which an IDM (maybe not a particularly well defined one) has attributes of entity data types that are needed in the IDM subset but which don't appear in the ADM. This is a typical STEP interpretation scenario, e.g., an AIM element has attributes .id and .name and the ARM application object that is mapped to it only has a name. IIDEAS needs to more strict about this – especially in terms of making any constraints on instantiation dependent on a subset/context – since our interest is in data *instance* reuse and sharing, not just data *specification* reuse and sharing as is the case for STEP.

The map between the IDM and each ADM that is integrated with it shall be recorded as part of the IDM.

4.3 Postconditions

This section and/or the clause that follows is misnamed – see comments under clause 6 regarding possible restructuring, in which case this would be *after* clause 4 in the document.

The overall activity of integrating an application data model is complete when the following specifications have been completed and validated:

— the subset of the IDM that precisely corresponds to the semantics of the ADM;

— the map that defines the transformations between the ADM and the identified IDM subset.

5 Preconditions

This may not be the best title for this clause – intent is to state the required and desirable characteristics of the inputs to the integration and mapping activities.

5.1 Application data model

The preconditions on an ADM are less stringent than those on other elements. The base level is that an ADM should be a data model (which I take here to be a documented data structure specification, that may include constraints) with a stated application purpose or viewpoint.

In the scenario illustrated by Figure 3 above, there is no ADM as an input: rather, one is created as a view on an IDM to satisfy a set of information requirements. Does this mean that we need a subclause stating the preconditions for "information requirements"? This is probably a good idea if we want to avoid the "implementable ARM" / "fully attributed ARM" issue that has afflicted STEP for many years. The question, though, is whether it is possible to define the criteria that distinguish an ADM from a less formally stated set of information requirements. I can certainly enumerate *some* of these, but I'm not confident at this time that a coherent and complete list of criteria can be generated.

5.2 Integration data model

I see this subclause as being a rewrite of the existing "high quality data models" material – the six principles. We may need to extend and or qualify this if we are making the methodology sufficiently general to cover the "integrate two ADMs" scenario (Figure 1 above), where its possible that not all the principles will apply. I know that we *can* argue for every IDM having a wide/universal context for reasons of reusability, integration, interoperability, etc. The question is whether we *should* require this approach within ISO 18876. This is the intent of the "modelling principles" component of the diagrams above, and we can use an example to illustrate whatever choice we make here. If I have two ADM fragments:

```
ENTITY part;
  part_name : STRING;
END_ENTITY;
```

```
ENTITY component;
  name : label;
  alternate_name : label;
END_ENTITY;
```

and analysis determines that **part** and **component** are really the same thing (probably **class_of_physical_object** and/or **physical_object** in EPISTLE terms), does the IDM that we only generalize ADM1 and ADM2 (two names, one being designated as "alternate" and being optional), or do we apply knowledge of the wider requirements for naming and allow for 0:? names with a specified role for each?

This is going to be a key issue especially if we hope that STEP will pick on IIDEAS as part of its evolution towards a future release of ISO 10303 based on new and improved IRs.

5.3 Languages

5.3.1 Modelling languages

This subclause will state the minimum requirements on modelling languages that can be used to specify ADMs and (particularly) IDMs. I take it, by the way, as a given within SC4 that EXPRESS is *by definition* sufficient to define at least some integration models. It will be interesting to see how long such as assumption survives. At some point we need to look at other candidate languages as well – EXIST (obviously!), the UML, KIF/CG, E-R variants, NIAM (does anyone use this any more?), XML Schema, …

In some respects this section should actually follow directly from what is in 5.1 and 5.2 – having stated the requirements on models in a language independent manner, the short version of this clause is that any modelling language should be capable of representing models that have those characteristics. This is also where the concept of the "overall model" comes in, as in a paradigm that splits "data model" and "data instance" rigidly the representation of the model may have to be a mixture of the two. For EXPRESS there may be a need to look at different types of instance representations, including EXPRESS constants and EXPRESS-I instances as well as "exchange file" representations such as Part 21 and Part 28.

5.3.2 Mapping languages

As with the previous section, the short form of this clause is relatively simple in that it needs to state that a suitable mapping language is capable of representing a map between an ADM and an IDM. However, there is clearly more involved here, as we need to consider a number of different capabilities of a mapping language to make assertions about the relationship between elements of two different models.

This will have to include (at least) the following (these are non-exclusive/overlapping):

- defining a map between two data types
- defining a map between a data type and a collection of data types

(These maps between data types may be misleading, as all maps really have to be defined in terms of instances. A map defined for a data type is just a short hand for a common map that applies to each and every instance of the data type – which I guess could itself be described in KIF or EXIST.)

- defining a map between a specified instance or collection of instances of a data type, and a data type (e.g., physical object classified as 'pump' in model A maps to pump in model B); this is probably more familiar expressed in the opposite direction, which is typical of ARM to AIM mappings in STEP APs
- encoding of data values (attribute transformations), e.g., model A has two attributes for person first name/last name, model B has just one name; issue here is whether this is recognized as a uni-directional mapping (cannot recover two values from one), or whether there is a possibility of specifying the grammar and therefore parsing the simpler form to extract the richer data set from it.
- conversion of data values (also attribute transformations) – this is the question raised by Robin La Fontaine at the May workshop in London. This is not really distinct from the previous type of mapping, but is worth mentioning in that the type of encoding/conversion may be different (or have different criteria applied). This will particularly be the case with anything numeric where accuracy and tolerance needs to be taken into account. A good example would be yards/metres conversions, and the appearance on UK roads of signs warning "Give Way 183m ahead".
- "parameterization" of mappings, i.e., providing a capability to specify common parts of maps once and reuse them. This has been developed as a refinement of the STEP AP mapping table syntax (not particularly well, but effective nonetheless). This seems to be absent from EXPRESS-X.

5.4 Services

We added this to the TOC at our last meeting in London – however, I'm not sure what the need is for this in Part 2. It seems to be a Part 1 component.

6 Integration and mapping methodology

In the original outline for this part this section (actually two sections, one titled "Integration method", the other "Mapping method", was intended to be the core of the specification and to contain all the details of the operation of the methodology. However, as the "overview" in clause 4 has grown considerably larger and more detailed than I originally envisaged, it is difficult at this time to see what additional detail could be given here that is both normative and independent of language/models.

Going back to the table of contents for Part 2 that was presented at the Melbourne meeting, we said that this part would cover several elements that probably end up as additional informative annexes. The pieces that we listed were:

Procedures ("what to do") – as far as is possible without dealing in the specific procedures associated with a given integration model, clause 4.2 goes a long way towards satisfying this requirement.

Practices – analysis ("how to do it") – again, 4.2 discusses this at a level of detail that is appropriate in an IDM-independent specification. However, if we follow the statements in clause 5.2 that lead (deliberately!) to the conclusion that EPISTLE-style models are well suited to act as IDMs, then we ought to be able to adapt parts of Developing High Quality Data Models as an informative annex on analysis practices.

Practices – languages and tools. Having established some preconditions for modelling and mapping languages in 5.3.1 and 5.3.2 there may not be much else to say here, unless we feel that it would be useful to provide some kind of simple "best practice guide" for some languages. Personally I think that this would be inappropriate here; such practices would have to be documented for an IDM defined in a given language.

Guidelines ("what's the best way to do this") – these were always intended to be outside the scope of the standard. Clearly one possible target is SC4 standing documents, although the drawback here is that all current standing documents are directed inward and really apply to SC4's own work. An alternative might be for some industrially-based body (EPISTLE, PDES Inc., …) to develop and publish Guidelines for use of a particular integration data model and, if there is a requirement this to have an ISO "label" of some kind, to submit it as a PAS.

7 Conformance

This section will define conformance requirements with respect to:

integration data models – referencing the preconditions documented in 5.2

integrated data models (i.e., an application data model with a defined/documented map to an integration data model.

A key test for this document is whether such conformance requirements can actually be documented and, if so, whether there is value to applying them. My assumption at this time is that any conformance requirements documented here will be "abstract" in nature serve as the basis for defining "real" conformance statements in other standards. For example, a standard integration data model can define conformance requirements on its extensions, or for the maps to be defined when application data models are integrated to/with it.

Annex A
(normative)

Information object registration

To provide for unambiguous identification of an information object in an open system, the object identifier

```
{iso standard 18876 part{2} version {1}}
```

is assigned to this part of ISO 15926. The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

NOTE This is the object identifier that will apply to the published (IS) version of this part of ISO 18876.

The reference to Part 1 of STEP assumes that the second edition of Part 1 defines usage of ASN.1 identifiers that is not limited to ISO 10303.

Annex B
(informative)

Technical discussions

B.1 Models, data, and data models

The purpose of this section is to provide additional clarification and examples of the terminology introduced in clause 3, and also to add further discussion on the concept of different representation forms for the same information.

Annex C
(informative)

Data model notation

In this International Standard a number of different data model notations are used with examples. These include ISO 10303-11 EXPRESS and ISO

This annex defines a simplified notation that is used in some of the examples in this International Standard. This notation is not intended as a substitute or replacement for languages such as EXPRESS or the UML; it is defined and used here in order to simplify examples and to support examples of mapping and integration.

C.1 Overview

The notation defined in this annex supports graphical representation of:

— entity data types;

— attributes;

— relationships (including subtype/supertype relationships);

— cardinalities.

It excludes:

— base data types (numbers, text strings);

— enumerated data types;

— constructed data types;

— constraints;

— entity data types with more than one supertype;

— subtypes with overlapping populations.

In the context of the architecture defined by ISO 18876-1, this notation is suitable for description of some simple application data models. It is not suitable for complex application data models, or for integration models.

NOTE Although the intent of this notation is limited to the examples within this International Standard, it can also be used to illustrate information requirements such as those that form the basis for the definition of an application view onto an integration model – see ????.

This note is saying – in a roundabout way! – that this notation is actually suitable for the definition of the ARMs of STEP APs, as they were originally envisioned within the STEP Architecture.

C.2 Entity data type

An entity data type is represented by a rectangular box; the name of the entity data type is written within the box. As this notation is intended for human understanding only, there is no restriction on the character set that may be used for the names of entity data types. An example of an entity data type in this notation is shown in Figure C.1 below.

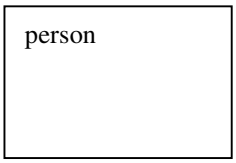

Figure C.1 – Graphical representation of an entity data type

C.3 Attributes

An attribute is represented by its identifier, preceded by a bullet, placed within the box that represents an entity data type. As with entity data types, there is no restriction on the character set that may be used for the names of attributes.

NOTE 1 The use of brackets '[' and ']' to delineate domain qualifiers for attributes, as described below, implies that their use within the names of attributes is potentially confusing, and therefore to be avoided.

An example of an entity data type with two attributes is shown in Figure C.2 below.

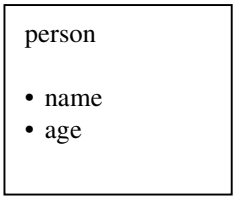

Figure C.2– Graphical representation of an entity data type with attributes

Although this notation does not formally include representation of the type or domain of an attribute, this may be indicated informally by qualifying the name of the attribute. The qualifier is placed after the name of the attribute and is delineated by brackets '[' and ']'. The word or phrase used to describe the domain of an attribute is not predetermined – the values used should either be self-evident or should be defined in accompanying text.

EXAMPLE 'real number' and 'text string' are possible values for domain qualifiers whose meaning does not require furether explanation.

An example of an entity data type with qualified attributes is shown in Figure C.3 below.

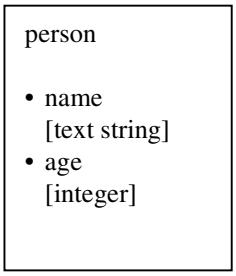

Figure C.3 – Graphical representation of an entity data type with qualified attributes

By default attributes are single valued and are mandatory. Different cardinalities may be specified by including the upper and lower limit on the number of values for an attribute in parentheses, separated by a colon. An example of an entity data type whose attributes have varying cardinalities is shown in Figure C.4 below.

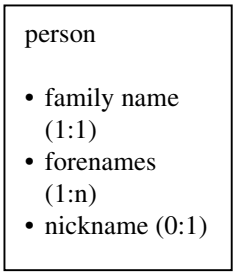

Figure C.4 – Graphical representation of attributes with non-default cardinalities

NOTE 2 The cardinality specified for attribute 1 is the explicit form of the default (single valued, mandatory) cardinality.

I am not really happy with the way in which cardinalities are handled here. My more natural inclination is to make the default "unspecified" (least constrained), which is the (0:n) case. The notation as described here shares the same failing as EXPRESS in this area, especially when we come to relationships and their inverses. On the other hand, readers/users of this notation may be surprised if an attribute that has no cardinality qualifier is potentially

NOTE 3 There is no equivalent in this notation of EXPRESS's OPTIONAL attributes, nor of distinctions between different types of aggregate attributes (ARRAY, BAG, LIST, SET).

NOTE 4 If required, both the domain and the cardinality of an attribute can be specified. This is illustrrated in ???? at the end of the annex.

C.4 Relationships

Relationships between entity data types are represented by a line drawn joining the boxes that represent the related entity data types. The line may be directed by placing an arrowhead at one end of the line: this indicates the direction in which the relationship should be read, and does not carry any implications of dependency or cardinality constraints.

An example of a relationship is shown in Figure C.5 below; the directed relationship indicates that this diagram should be read as "person owns car" and not "car owns person".

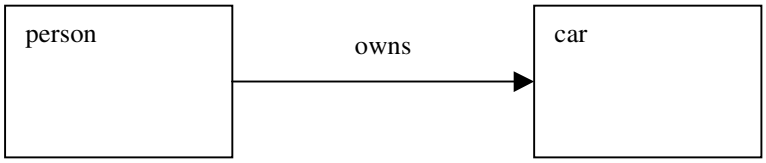

Figure C.5 – Graphical representation of a relationship between two entity data types

By default relationships are single valued and are mandatory. Different cardinalities may be specified by including the upper and lower limit on the number of values for a relationship in parentheses, separated by a colon. This qualifier is placed adjacent to the entity data type to which the cardinality constraint applies. An example of a relationship with specified cardinalities is shown in Figure C.6 below.

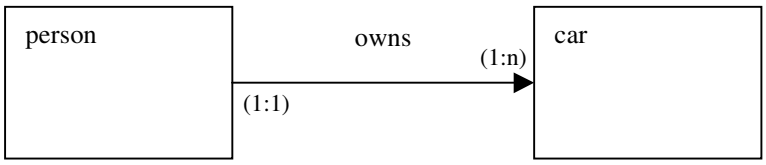

Figure C.6 – Graphical representation of a relationship with specified cardinalities

The cardinality constraints here should be read as:

— each person owns one or more cars;

— each car participates in exactly one relationship with person.

All relationships are bi-directional. In this notation different names can be assigned to each direction in the relationship; in this case, the directed (arrow) notation used above is ambiguous. The correct association between a relationship name and a direction can be specified in two ways:

— by placing the relationship name closer to the entity data type box that is the start of the relationship, or

— by placing the relationship name and its cardinality constraint on the same side of the relationship line.

These two approaches are illustrated in Figure C.7 and Figure C.8 below.

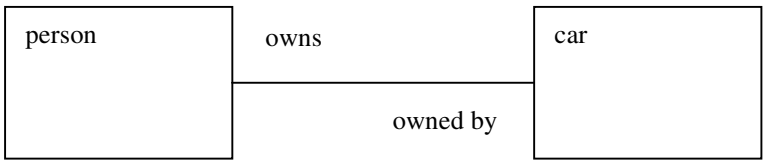

Figure C.7 – Graphical representation of a relationship that is named in both directions

Figure C.7 should be read as:

— each person owns exactly one car;

— each car is owned by exactly one person.

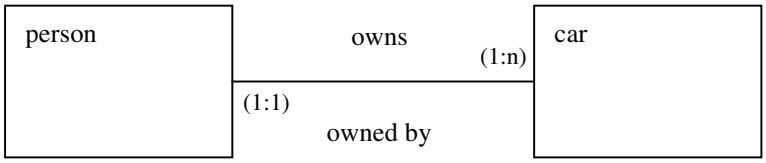

Figure C.8– Graphical representation of a doubly-named relationship with specified cardinalities

Figure C.8 should be read as:

— each person owns one or more cars;

— each car is owned by exactly one person.

C.5 Subtypes and supertypes

Representation of subtype/supertype (specialization/generalization) is supported by this notation by placing the box that represents an entity data type inside the box that represents its supertype. An example of a sub-type/supertype relationship is shown in Figure C.9 below.

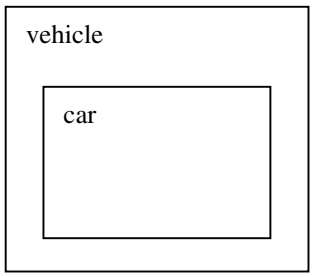

Figure C.9 – Graphical representation of a subtype/supertype relationship between two entity data types

An entity data type can have several subtypes. This notation does not consider the relationships between sub-types; in Figure C.10 below:

— each car is a vehicle;

— each truck is a vehicle;

— each vehicle can be a car;

— each vehicle can be truck.

No further information on the possible combinations can be inferred from this notation. Textual description of an entity type may add constraints; in the example shown in Figure C.10, it could be stated that:

— each vehicle is either a car or a truck;

— no vehicle is both a car and a truck.

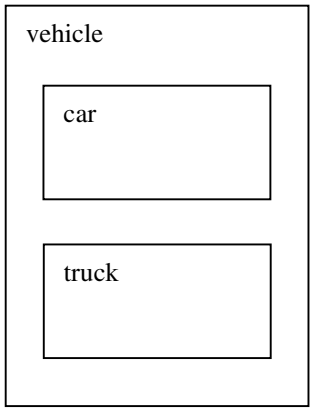

Figure C.10 – Graphical representation of an entity data type with two subtypes

C.6 Example using the notation

An example data model using this notation is shown in Figure C.11 below.

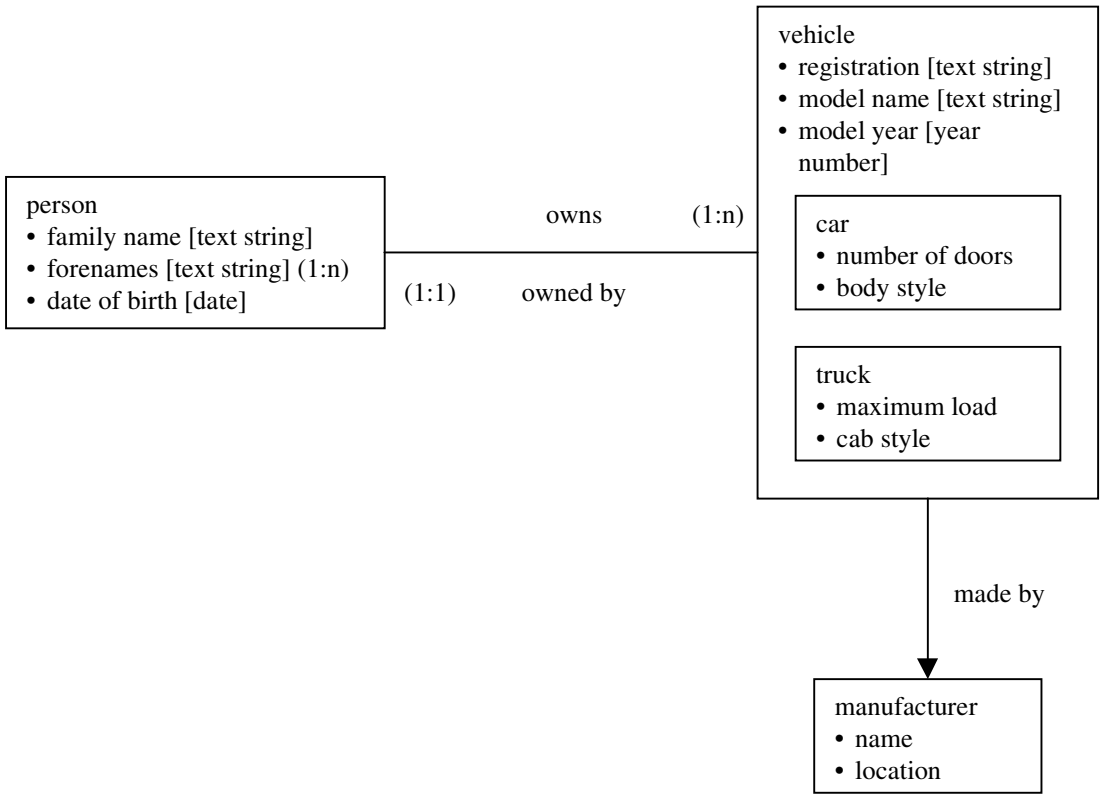

Figure C.11 – Example data model

NOTE As stated in C.3 above, a narrative description of this model should explain any domain qualifiers that are not self-evident. The meaning of entity data types, attributes, and relationships should also be provided.

Annex D
(informative)

Worked examples

This may be a large component of the document. Several proposals have been made for the basis of such examples:

- integrating AP203 and AP210 (possibly in the area dealing with product structure)
- integrating AP232 and the PDM schema/modules
- defining application views onto ISO 15926-2, using the EPISTLE work on “templates” as a starting point
- integrating PLIB’s dictionary schema with ISO 15926-2
- integrating two XML DTDs with overlap in their scope/subject area

Only the last of these really goes outside the SC4/EXPRESS modelling domain – we probably need to identify some examples from outside – one possibility that comes to mind is to take the IAI’s geometry model (which uses EXPRESS but assumes a default ONE OF and, although based on Part 42 of STEP is different from it), and the UML geometry model produced by TC211 in the GIS domain.

Another possibility might be something from the EDIF area – possibly Hilary Kahn and/or Robin La Fontaine could supply something here based on the work of the STEPWISE project.

For the initial drafts at least the number of examples to be included will have to be restricted,

Bibliography

This is the content of the Bibliography of ISO 15926-1. Most of the documents listed seem relevant here too!

1. American National Standards Institute. *ANSI/X3/SPARC Study Group on Database Management Systems Interim Report*, 1975.

2. ANGUS, Chris; DZIULKA, Peter. *EPISTLE Framework.* Version 2.0, Issue 1.21. EPISTLE, April 1998. Available from http://www.stepcom.ncl.ac.uk.

3. *Integration Definition for Information Modeling (IDEF1X).* Federal Information Processing Standards Publication 184, December 1993

4. ISO/CD 10303-231, *Industrial automation systems and integration — Product data representation and exchange — Part 231: Application protocol: Process engineering data: Process design and process specifications of major equipment.* ISO TC184/SC4/WG3 N745, 1998-11-24.

5. Petrotechnical Open Software Corporation. *Data Access And Exchange*. PTR Prentice-Hall, Inc., Englewood Cliffs, 1994.

6. POSC/CAESAR Project, *Oil & Gas Facilities Data Model.* Available from http://www.posccaesar.org.

7. POSC/CAESAR Project, *Oil & Gas Facilities Reference Data Library.* Available from http://www.posccaesar.org.

8. PROCESS INDUSTRIES STEP CONSORTIUM. *STEP In The Process Industries: Process Plant Engineering Activity Model*. Issue 1, July 1994.

9. WEST, Matthew; FOWLER, Julian. *Developing High Quality Data Models*. Version 2.0, Issue 2.1. EPISTLE, 1996. Available from http://www.stepcom.ncl.ac.uk.

Index

